Travis Xie, 1864724

Part_B_Report

In the handle_transition function, I implemented the normal version of Q update to update the Q_Value for the previous state and update the previous state to the current state. Then the choose_next_action function will call the handle transition function first to update the Q_Value if it is not the initial state. Next, it will choose next action to play. If the state is the goal state, the next action will be exit with no doubt. If the state is terminal, the next action will be None. Otherwise, the function will use either greedy epsilon method or the exploration function to select the next action. In the greedy epsilon method, in addition to the pre-decided epsilon value, I also implemented a custom epsilon value that follows an exponential distribution.

$$Pr(x) = \lambda e^{-\lambda \frac{x}{10^n}}$$

where n is the number of disks, x is the number of times that Q-Value changes, and $\lambda = 0.8$. The use of n value is to allow the probability to converge with different speeds in different games. In general, the probability of exploring will decrease as x increases, but I specially set a lower bound value of epsilon to 0.2 to prevent the agent from being stuck either in a sub-optimal policy or in an invalid action. Then agent will choose either explore or exploit.

For exploration function, I took the inspiration from the lecture slides to make it as following

$$f(Q, N) = Q(s, a) + \frac{K}{N(s, a)}$$

Where K is an arbitrary value that controls the speed of convergence, and $N(s, a)$ is the number of times that a pair of (state, action) is chosen. Next action will be the action that maximizes $f(Q, N)$. At the beginning of the learning, the agent is more likely to explore the less-visited states, but as some good Q_Values being discovered, the agent will begin to stick with the current policy.

When playing with the agent, I found out that in general, the smaller the epsilon value is, the more training the agent needs to find the optimal policy because, with a small epsilon value, the agent is more likely to exploit the current policy. Once the agent gets stuck within either an invalid action or a suboptimal policy, it will not come out until a noise or an exploration occurs. However, with the custom epsilon and exploration function, the agent will explore as much as states at first and then change to exploit as the value converges. So, they will take less training to find the golden path, but it is really difficult to tell which one, the custom or the exploration function, performs better since there are not enough samples to do so.